Proceedings of IDETC/CIE 2005
ASME 2005 International Design Engineering Technical Conferences
& Computers and Information in Engineering Conference
September 24-28, 2005, Long Beach, California USA

DETC2005-84699

COLLISION DETECTION OF CYLINDRICAL RIGID BODIES USING LINE GEOMETRY

John S. Ketchel
Robotics and Spatial Systems Laboratory
Mechanical and Aerospace Engineering Department
Florida Institute of Technology
Melbourne, Florida 32901
Email: jketchel@fit.edu

Pierre M. Larochelle*
Robotics and Spatial Systems Laboratory
Mechanical and Aerospace Engineering Department
Florida Institute of Technology
Melbourne, Florida 32901
Email: pierrel@fit.edu

ABSTRACT

This paper presents a novel methodology for detecting collisions of cylindrically shaped rigid bodies moving in three dimensions. This algorithm uses line geometry and dual number algebra to exploit the geometry of right circular cylindrical objects to facilitate the detection of collisions. First, the rigid bodies are modelled with infinite length cylinders and a necessary condition for collision is evaluated. If the necessary condition is not satisfied then the two bodies are not capable of collision. If the necessary condition is satisfied then a collision between the bodies may occur and we proceed to the next stage of the algorithm. In the second stage the bodies are modelled with finite length cylinders and a definitive necessary and sufficient collision detection algorithm is employed. The result is a straight-forward and efficient means of detecting collisions of cylindrically shaped bodies moving in three dimensions. This methodology has applications in spatial mechanism design, robot motion planning, workspace analysis of parallel kinematic machines such as Stewart-Gough platforms, nuclear physics, medical research, computer graphics and well drilling. A case study examining a spatial 4C robotic mechanism for self collisions is included.

INTRODUCTION

In this paper we present an algorithm for determining quantitatively if two bodies moving in three dimensional space collide. The methodology presented consists of two stages. In the first, infinite length cylinders are used to model the objects, then line geometry is used to determine if the cylinders intersect. If these infinite cylinders do not intersect then the two bodies do not collide and no further testing is required. If the two infinite cylinders do intersect then further testing is necessary. We proceed to the second stage where cylinders of finite length are used to model the objects and they are tested to determine quantitatively if they collide.

Collision detection is vital for real world implementation of three dimensional mechanical systems such as robots, mechanisms, parallel kinematic machines, and linkages. Collision detection assists in motion planning, digital prototyping and motion simulation of the system. For motion planning applications collision detection can be used to verify that the planned motion of the system is collision-free with respect to the working environment and self-collisions. The methodology presented here enables the user to model the system and determine *without risking hardware* if there is a possible collision. Three dimensional mechanical systems can be difficult and expensive to develop hence such modelling and testing of the system in the early stages of design may save time and reduce costs. Often, complex systems are digitally prototyped and simulated. These simulations are improved by including motion planning and collision detection.

The methodology presented here is general and can be used to detect collisions between any rigid bodies moving in three dimensions provided that the bodies are predominantly cylindrical

*Address all correspondence to this author.

in shape. We focus upon such bodies because this shape is commonly found in industrial robots, parallel kinematic machines (e.g. Stewart-Gough platforms), and spatial mechanisms. Our primary motivation for this work comes from our efforts to advance the state of the art in spatial mechanism design. Spatial mechanisms are closed kinematic chains consisting of rigid links connected by cylindric(C), revolute(R), or prismatic(P) joints. Traditionally, the links of these mechanisms are cylindrical in shape. Recently, there have been some significant efforts made to address the challenge of designing useful spatial mechanisms. In [9] Larochelle presented a Burmester Theory based computer-aided design program for spatial 4C mechanisms. Efforts were made to address circuit and branch defects in [8]. Approximate motion synthesis was addressed by [10] and [2]. The exploration of utilizing virtual reality techniques to address the inherent visualization and interaction challenges was reported in [7] and [11].

The goal here was to facilitate the design of spatial 4C robotic mechanisms by assisting in the selection of a mechanism that is free of collisions, including self-collisions. For collision detection, cylinders are used to model all objects that need to be analyzed. Cylinders are useful modelling tools since most three dimensional mechanical systems consist of some combination of prismatic, revolute, and cylindric joints. With respect to the specific case of spatial 4C robotic mechanisms, traditionally the links are cylindrical in shape. For the first stage of testing, infinite length cylinders are used to model the links. This allows us to employ line geometry to yield a fast and efficient means of determining if a collision is possible. If the infinite cylinders do intersect then the actual finite cylinders may in fact collide. Therefore, we proceed to the second stage of the collision detection algorithm where finite length cylinders are used to model the links. Then these cylinders are tested rigorously for possible collisions.

The paper proceeds as follows. First, the distance calculations between infinite cylinders, then finite length cylinders are presented. The necessary kinematic analysis of the spatial 4C mechanism are performed. Next, utilizing the distance calculations and the results of the spatial 4C analysis, we determine if a collision occurs for a spatial 4C mechanism. Finally, a case study for the self collision detection of a spatial 4C mechanism is presented.

Related Works

Collision detection is vital for real world implementation of three dimensional systems such as spatial mechanism design, mobile and autonomous robot motion planning, workspace analysis of parallel kinematic machines such as Stewart-Gough platforms, nuclear physics, computer graphics, well drilling, Mars rovers and medical research. Collision detection assists in motion planning, real-time control, digital prototyping and motion simulation of these systems.

Zsombor-Murray [18] presents the visualization of the shortest distance between two lines in space. His constructive geometry and algebraic solutions to the problem motivated the work proposed here. [17] correctly states that failure to detect a collision is less acceptable than false positives, which can be further checked and that for the sake of speed exact or accurate collision detection is often sacrificed.

Collisions are unacceptable and being able to detect and avoid them is of vital interest. A great amount of prior work has been done on the collision detection problem that has resulted in the developed of many software packages such as: VEGAS, V-Clip, RAPID, SOLID, I-Collide, V-Collide and PQP. They vary in their modelling and in mathematical method for determining if a collision has occurred.

Virtual Environment for General ASsembly (VEGAS) [22] is a fully immersive virtual environment that permits the user to explore various assembly situations. It uses triangles whose vertices can be added together to generate polygons. The polygons are transformed into small cubes (voxels) to model the entire environment so that their Voxmap PointShell method (VPS) can be utilized. VPS uses surface normals to calculate reaction forces to determine if a collision has occurred. VPS was designed for fast, not accurate collision detection.

Voronoi Clip (V-Clip) [20] uses convex polyhedrons to model the system and then tracks each polyhedrons features (vertices, edges and faces). V-Clip tracks the closest features between each pair of polyhedrons. Knowing which feature is closest makes calculating the distance between them easy. It uses existing libraries (Qhull) to build its hierarchies of convex polyhedrons.

RAPID [20] is based on two different algorithms that uses Oriented Bounding Boxes (OBBs) to model the system. The first uses a top-down decomposition technique that builds an OBB hierarchy or OBBtree. The second one tests for collisions between the OBB pairs. It tests if the higher level OBBs overlap which indicates a possible collision and the next lower level needs to be tested. RAPID uses fifteen simple axis projection tests to determine if a collision has occurred.

Similarly, SOLID [20] also uses two algorithms for it's collision detection methods. First, it creates a bounded hierarchal volume composed of Axis-Aligned Bounding Boxes (AABB). Second, it computes the distance between two convex polytypes using the Minkowski difference and convex optimization techniques.

I-Collide [19] also uses a two-level approach for collision detection. It is based on a model comprised of multiple levels of bounding boxes. I-Collide removes object pairs from the model using exact collision testing between the pairs of polyhedra. It tracks the closest points between the pairs of convex polytypes. It can also create a convex polytype model tree for non-convex objects.

V-Collide [21] uses two existing collision detection libraries.

Initially it uses I-Collide to determine the possible collisions among a large number of objects. It then uses RAPID to perform pairwise testing to determine if a collision has actually occurred.

PQP [20] creates a hierarchy of Rectangle Swept Spheres (RSS), volumes covered by a sphere whose center is swept over a 3D rectangle. It uses a specialized algorithm to improve the efficiency and robustness of the distance calculations. Distance calculations are performed between the RSSs on the hierarchal tree.These general methods are computationally intensive when compared to the algorithm presented here.

Although spatial 4C robotic mechanisms are capable of spatial motion, i.e. motion in three dimensions, their motion is constrained to a complex three dimensional surface so traditional methods of path planning (e.g. [1]) do not apply. Similarly, singular configurations can be easily identified during the testing phase and can be avoided during implementation [13]. Path verification [16] and reachable space methods [6] do not apply for the self collision problem since the envelope of the mechanism does not take into account self collisions. There are several methods available to calculate the data necessary for the two step analysis that is proposed in this paper. The relatively low number of points and vectors necessary used do not warrant using vector bundles [15].

COLLISION DETECTION

Infinite Cylinder Testing

Here we use infinite and finite length cylinders to model rigid bodies in three dimensions. Initially, each object is modelled by a cylinder of infinite length and finite radius. Infinite cylinders are simple models to check for collisions since they can be represented as a line in space with a radius. The shortest distance between two lines in space is along their common normal line N. An advantage of using cylinders is that their common normal line has a finite line segment between the two cylinders and by subtracting the two cylinders radii from the segment, possible collisions can be detected. If the distance between the two cylinders is less than the sum of their two radii then the infinite cylinders have collided. Hence, if the actual finite cylindrical objects have collided it is *necessary* that the minimum distance between their associated infinite cylinders be less than the sum of their radii.

The major axis of an infinite cylinder is a line. Here, we use Plücker coordinates and dual vectors to represent these lines in space. Plücker coordinates define a line by its unit directional vector and moment. Moreover, when convenient, we employ dual vector algebra to operate on lines. The Plücker coordinates of a line can be generated from two points on the line or from a point and direction vector (see fig. 1). For example line S_1 can be defined by points $\vec{c}$ and $\vec{f}$ or point $\vec{c}$ and direction vector $\vec{s}$ (see eqs. 1 and 2).

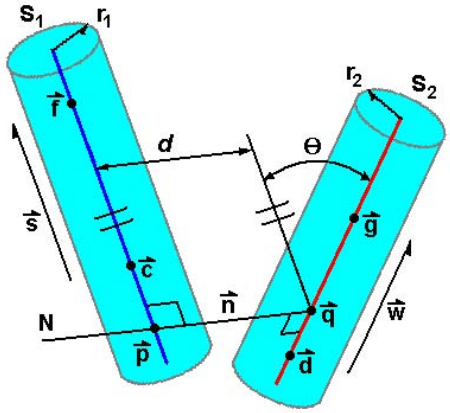

Figure 1. Infinite Cylinders

$$S_1 = \left(\frac{\vec{f}-\vec{c}}{\|\vec{f}-\vec{c}\|}, \vec{c}\times\frac{\vec{f}-\vec{c}}{\|\vec{f}-\vec{c}\|}\right) = (\vec{s}, \vec{c}\times\vec{s}) \tag{1}$$

$$S_2 = \left(\frac{\vec{g}-\vec{d}}{\|\vec{g}-\vec{d}\|}, \vec{d}\times\frac{\vec{g}-\vec{d}}{\|\vec{g}-\vec{d}\|}\right) = (\vec{w}, \vec{d}\times\vec{w}) \tag{2}$$

We use the dual vector representation of the lines and dual vector algebra as follows [4] and [12] where ε^2=0.

$$\hat{S}_1 = (\vec{s}, \vec{c}\times\vec{s}) = (a, a^0) = a + \varepsilon a^0 \tag{3}$$

$$\hat{S}_2 = (\vec{w}, \vec{d}\times\vec{w}) = (b, b^0) = b + \varepsilon \vec{b}^0 \tag{4}$$

Line dot product:

$$\begin{aligned}\hat{S}_1 \cdot \hat{S}_2 &= (a, a^0)\cdot(b, b^0) \\ &= (a\cdot b, a\cdot b^0 + b\cdot a^0) \\ &= a\cdot b + \varepsilon(a\cdot b^0 + b\cdot a^0) \\ &= \cos\theta - \varepsilon d\sin\theta \\ &= \cos\hat{\theta}\end{aligned} \tag{5}$$

Line cross product:

$$\begin{aligned}\hat{S}_1 \times \hat{S}_2 &= (a, a^0) \times (b, b^0) \\ &= (a \times b, a \times b^0 + a^0 \times b) \\ &= a \times b + \varepsilon(a \times b^0 + a^0 \times b) \\ &= (\sin\theta + \varepsilon d \cos\theta)\hat{N} \\ &= \sin\hat{\theta}\hat{N}\end{aligned} \tag{6}$$

where $\hat{N}$ is the common normal line to $\hat{S}_1$ and $\hat{S}_2$.

The above operations are useful for calculating the distance d and the angle θ between two lines. The resultant dual number of the dot product of two dual vectors yields the angle and distance between the two lines as long as they are not parallel to each other (see eq. 5). If the $d\sin\theta$ term is not equal to zero, then the lines do not intersect ($d \neq 0$) and are not parallel ($\sin\theta \neq 0$). If $d\sin\theta$ is equal to zero and the $\cos\theta$ term does not equal one, then the lines intersect (d=0) and are not parallel. If the $\cos\theta$ term of the dot product is equal to 1 then the lines are parallel and the resultant dual vector of the cross product will have a 0 real component. The cross product's dual component ($d\cos\theta$) will be 0 when the lines are identical. If the $d\cos\theta$ term is non-zero then the distance, d, can be calculated. Figure 2 shows a detailed flow chart of the infinite cylinder test procedure.

This is an efficient method of determining the distance between the two lines and if a possible collision has occurred. If the resulting distance is greater than the sum of the two radii then no collision is possible regardless of the length of the finite cylinders. If the result is not greater than the sum of the two radii then a collision *may have* occurred and a finite cylinder model is used in the next stage of the collision detection algorithm.

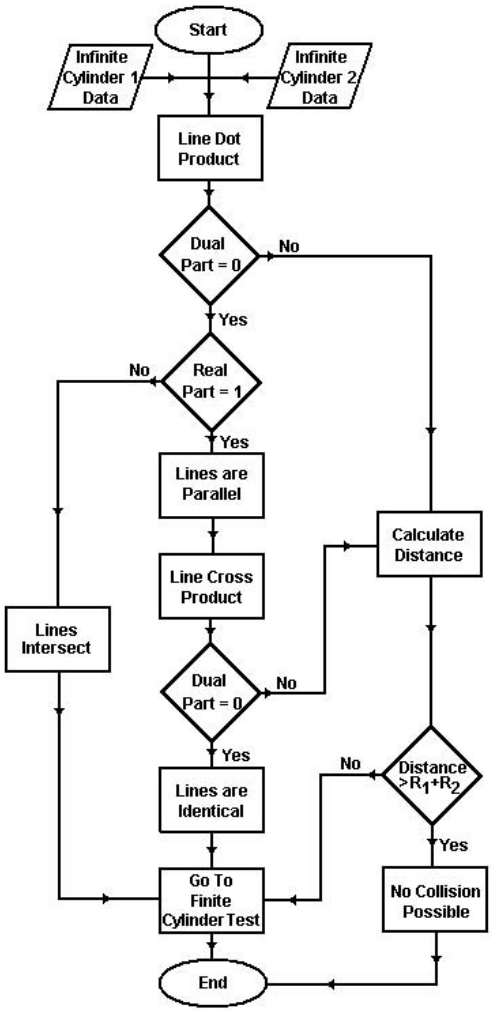

Figure 2. Infinite Cylinder Testing

Finite Cylinder Testing

If a possible collision has been detected by the infinite cylinder test then further testing is required to determine if an actual collision has occurred. The model is modified from cylinders of infinite length to cylinders of finite length. This changes the approach from testing lines to testing line segments. The same idea applies that the shortest distance between the cylinders is their common normal but the point where the common normal intersects the cylinder's axis becomes important. Figure 3 shows a detailed flow chart of the initial testing of the finite cylinders.

Parallel Testing From the initial testing, the angle θ between the infinite cylinders is known. If the finite cylinders are parallel then there are two general cases: their associated line segments overlap in some manner or there is no overlap. To determine if the segments overlap the plane that is orthogonal to the lines and passes through one endpoint is determined. The intersection point of this plane with the other line is computed. This point is then tested to see if it is on the line segment or not. This is repeated for up to two other endpoints (a fourth endpoint being redundant) to test for overlapping. When no overlapping occurs, no collision is possible.

Non-Parallel Testing To facilitate the testing of the non-parallel finite cylinders we require that the angle between their axes be acute. If the angle is not acute, then the endpoints of the second finite cylinder are interchanged guaranteeing that the angle between the axes is acute.

The shortest distance between two non-parallel lines is along their common normal. We begin by determining where the common normal line intersects the axis of each cylinder. The axes are described by their endpoints ($\vec{c}$ and $\vec{d}$) and *non*-unit direction vectors ($\vec{s}$ and $\vec{w}$), where $\|\vec{s}\|$ and $\|\vec{w}\|$ are equal to the length of their corresponding cylinder (see fig. 4). The common normal

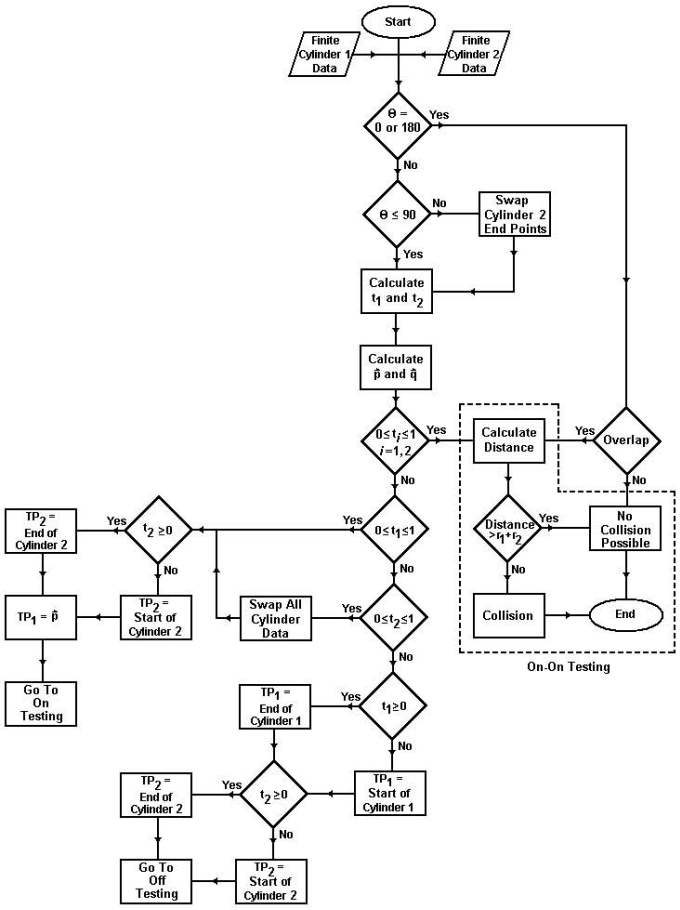

Figure 3. Finite Cylinder Testing

line N intersects the lines S_1 and S_2 at points $\vec{p}$ and $\vec{q}$ respectively. Parametric equations for points $\vec{p}$ and $\vec{q}$ of lines S_1 and S_2 are:

$$\vec{p} = \vec{c} + t_1\vec{s} \qquad (7)$$
$$\vec{q} = \vec{d} + t_2\vec{w}$$

where

$$t_1 = \frac{[(\vec{d}-\vec{c})\times\vec{w}]\cdot\vec{n}}{\vec{n}\cdot\vec{n}}$$
$$t_2 = \frac{[(\vec{d}-\vec{c})\times\vec{s}]\cdot\vec{n}}{\vec{n}\cdot\vec{n}}$$

and

$$\vec{n} = \vec{s}\times\vec{w}$$

Next for each cylinder we need to determine the test point along it's axis, within the finite cylinder, that is closest to the common normal line. These test points are referred to as TP_1 and TP_2. Then we determine if common normal points, $\vec{p}$ and $\vec{q}$, are on the segments, before the segments or after the segments.

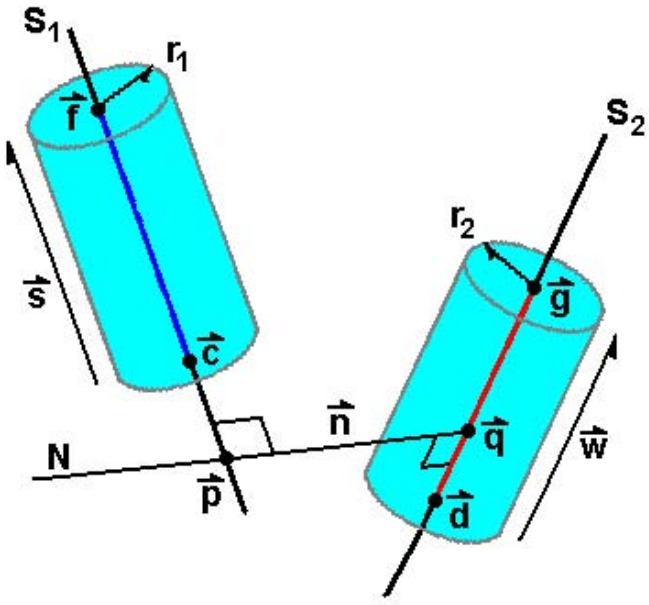

Figure 4. Finite Cylinders

If $t_1 \leq 0$ then $\vec{p}$ lies at the start of the segment or earlier, so the start point of the cylinder is used as TP_1. If $t_1 \geq 1$ then $\vec{p}$ lies at the end of the segment or further, so the end point is used as TP_1. If $0 < t_1 < 1$ then $\vec{p}$ lies on the line segment so $\vec{p}$ can be used as TP_1. The above procedure is repeated for cylinder 2 to determine TP_2.

From the determination of whether points $\vec{p}$ and $\vec{q}$ lie on or off the cylinders there are three possible cases to consider. If $0 \leq t_i \leq 1$, where $i = 1,2$, both $\vec{p}$ and $\vec{q}$ lie on the segments and On-On testing is necessary. If either but not both $\vec{p}$ and $\vec{q}$ lie on the segments, On testing is necessary. If neither $\vec{p}$ nor $\vec{q}$ lie on the segments, Off testing is necessary. These cases are discussed below.

Case 1 - On-On Testing The distance between the test points can be compared to the sum of the radii of the cylinders to determine if a collision has occurred. No further testing is required.

Case 2 - On Testing Figure 5 shows a detailed flow chart of the finite cylinder On test procedure. If only one cylinder is On, the testing requires that it is cylinder 1. If it is cylinder 2 that is On, all cylinder 1 and 2 data is swapped. Next, we find the closest point along cylinder 1's axis to TP_2 (see fig. 6). This point, $\vec{p}_1$, is the intersection of lines S_1 and N_1 (N_1 is orthogonal to S_1 and passes through TP_2). Calculating $\vec{p}_1$ (see eqs. 8) yields t_3 which is used to determine if $\vec{p}_1$ lies on or off the cylinder. If the distance from TP_2 to $\vec{p}_1$ is greater than the sum of the radii then no collision is possible and no further testing is required.

Otherwise, we must determine if $\vec{p}_1$ lies on or off cylinder 1. If the distance from the closest end of cylinder 1 to $\vec{p}_1$ is less than r_2 then further testing is required. The closest point (TP_2') of cylinder 2 to cylinder 1's axis is on the circular end of cylinder 2. TP_2' can be found by adding r_2 in the $\vec{n}$ direction to TP_2 (see eqs. 9). TP_2' is on cylinder 2's end circle. Next, we find the closest point along cylinder 1's axis to TP_2'. This point, $\vec{p_1'}$, is the intersection of lines S_1 and N_1' (N_1' is orthogonal to S_1 and passes through TP_2'). Calculating $\vec{p_1'}$ (see eqs. 10) yields t_3' which is used to determine if $\vec{p_1'}$ lies on or off the cylinder. If both p_1 and p_1' lie off cylinder 1 then no collision is possible and no further testing is required. If only p_1 lies on cylinder 1 then the ends of the cylinders may intersect (see fig. 7) and end intersection testing (discussed below) is required. If only p_1' lies on cylinder 1 (see fig. 8) or both p_1 and p_1' lie on cylinder 1 then the distance is calculated from TP_2' and p_1'. If the distance is less than cylinder 1's radius a collision has occurred. If it is greater then no collision is possible.

$$\vec{p}_1 = \vec{c} + t_3\vec{s} \tag{8}$$
$$\vec{s}\cdot\vec{p}_1 = \vec{s}\cdot TP_2$$
$$TP_2' = TP_2 + r_2\vec{n} \tag{9}$$
$$\vec{n} = \frac{\vec{p}-\vec{q}}{\|\vec{p}-\vec{q}\|}$$
$$\vec{p_1'} = \vec{c} + t_3'\vec{s} \tag{10}$$
$$\vec{s}\cdot\vec{p_1'} = \vec{s}\cdot TP_2'$$

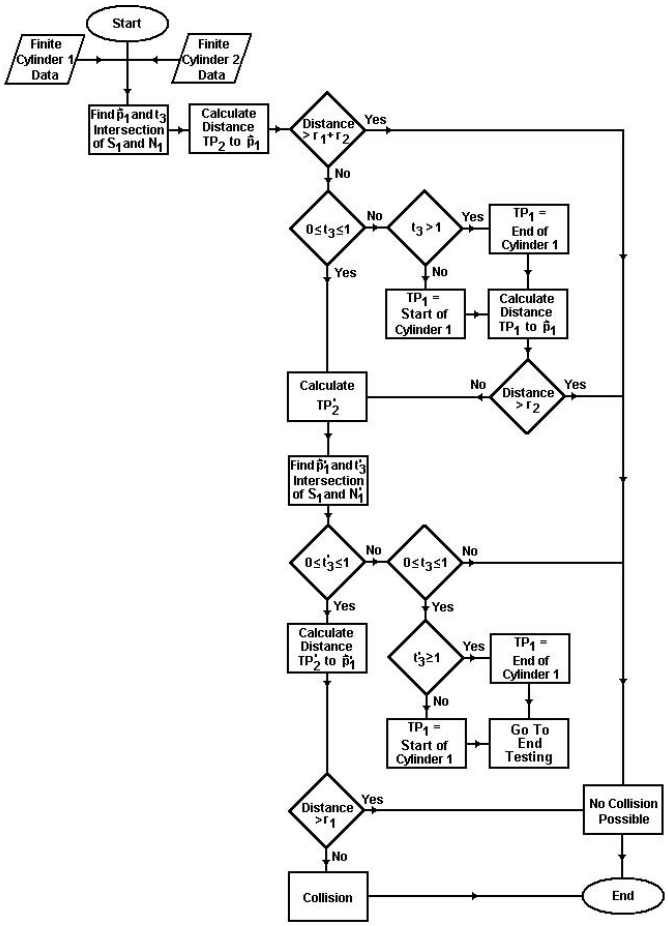

Figure 5. On Testing

Case 3 - Off Testing Figure 9 shows a detailed flow chart of the finite cylinder Off test procedure. This case has multiple test points that need to be tested. For each point, it uses the same testing as the On case. The distance from TP_2 to S_1 is found. If the distance is greater than the sum of the radii then no collision is possible since TP_2 is the closest point on cylinder 2 to cylinder 1 thus completing cylinder 2 testing. If the distance is not greater than the sum of the radii then On testing must be performed. Additionally, the other end of cylinder 2 must be similarly tested. The cylinder data is swapped and the Off testing is repeated with the new data.

End Testing Figure 10 shows a detailed flow chart of the finite cylinder end test procedure. End cylinder testing is necessary when the circular ends of the cylinders may intersect (see fig. 11). It is possible for the test points, when the projections are added, to project on or off the cylinder. When this occurs it is necessary to check if the cylinder ends intersect resulting in a collision. The approach used to test for collisions uses the distance from the line that is the intersection of the planes of the cylinder ends to the test points. The first step is to find the equations of the planes, π_1 and π_2, that are orthogonal to each cylinder's axis and pass through their test points (see eqs. 11).

$$\pi_1 : \vec{s}\cdot\begin{bmatrix}x\\y\\z\end{bmatrix} = \vec{s}\cdot TP_1 \tag{11}$$
$$\pi_2 : \vec{w}\cdot\begin{bmatrix}x\\y\\z\end{bmatrix} = \vec{w}\cdot TP_2$$

Then the parametric equation for the line of intersection M of π_1 and π_2 is found (see eq. 12). This is done by setting either x, y or $z = 0$ and solving simultaneously for the remaining components.

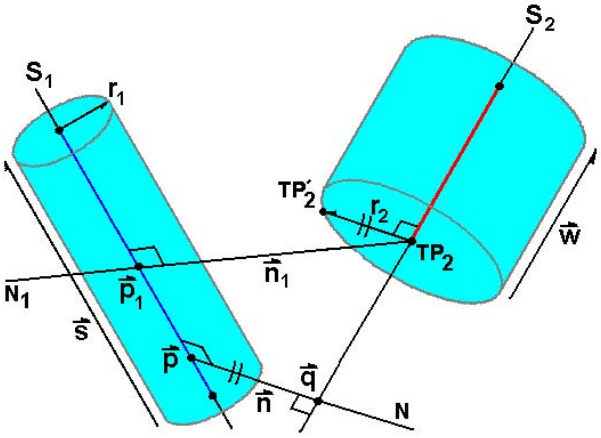

Figure 6. On Testing Labels

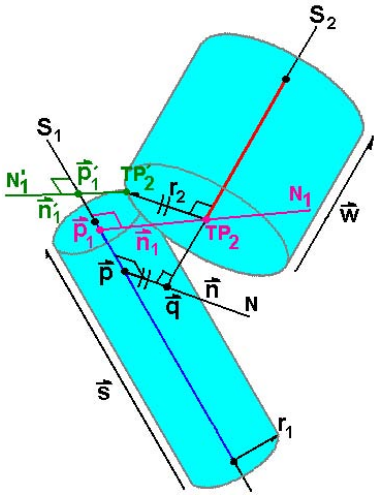

Figure 7. Projecting Off

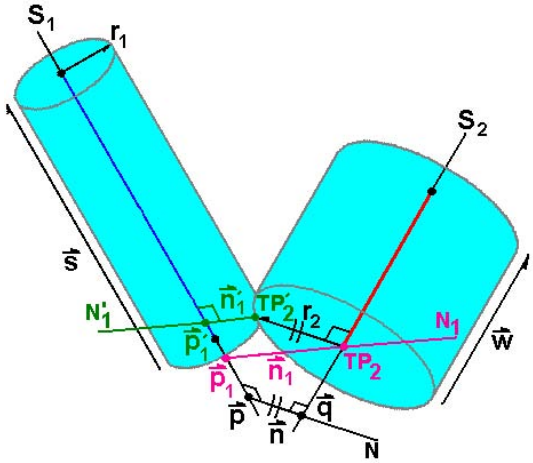

Figure 8. Projecting On

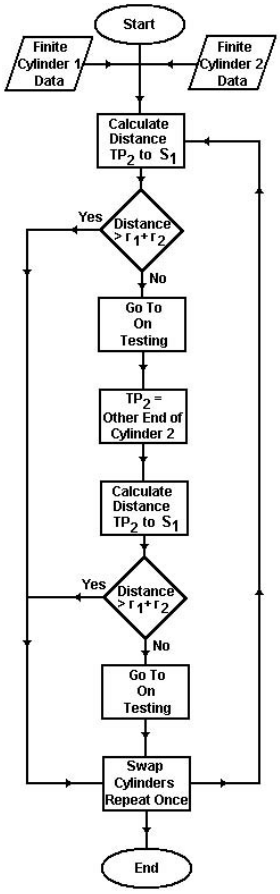

Figure 9. Off Testing

$$M = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + t\vec{m} \tag{12}$$

Next the distance from TP_1 to M is set to r_1 to obtain a quadratic in t_4:

$$r_1^2 = \|M - TP_1\| \tag{13}$$

$$r_1^2 = \| \begin{bmatrix} x \\ y \\ z \end{bmatrix} + t_4\vec{m} - TP_1 \|$$

$$0 = t_4^2(m_x^2 + m_y^2 + m_z^2) + t_4(2m_x(x - TP_{1x}) + 2m_y(y - TP_{1y}) + 2m_z(z - TP_{1z})) + ((x - TP_{1x})^2 + (y - TP_{1y})^2 + (z - TP_{1z})^2 - r_1^2)$$

Similarly, t_5 can be found by setting the distance from TP_2

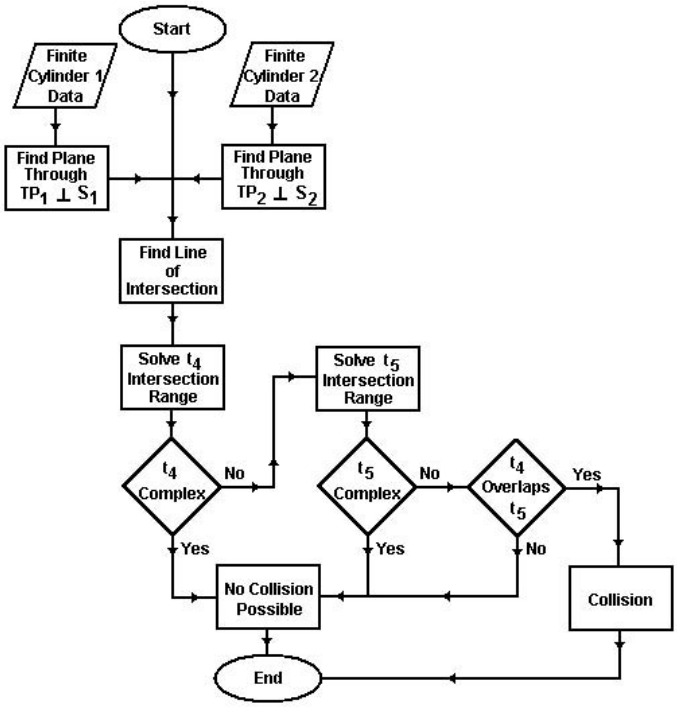

Figure 10. End Testing

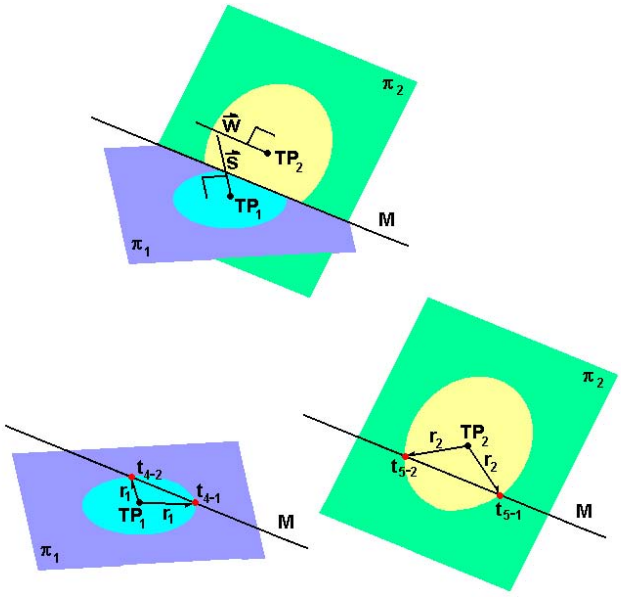

Figure 11. Cylinder End Overlap

to M as r_2 (see eqs. 14).

$$r_2^2 = \|M - TP_2\| \tag{14}$$

$$r_2^2 = \| \begin{bmatrix} x \\ y \\ z \end{bmatrix} + t_5\vec{m} - TP_2\|$$

$$0 = t_5^2(m_x^2 + m_y^2 + m_z^2) + t_5(2m_x(x - TP_{2x}) + 2m_y(y - TP_{2y}) + 2m_z(z - TP_{2z})) + ((x - TP_{2x})^2 + (y - TP_{2y})^2 + (z - TP_{2z})^2 - r_2^2)$$

If the quadratic yields complex roots then the cylinder end circle and the line M do not intersect and no collision is possible. Repeated roots from the quadratic formula mean that the cylinder end circle is tangent to the line M and no collision has occurred. If both roots are real then a range is found for t_4 and t_5. If the t_4 and t_5 ranges overlap then a collision has occurred.

THE SPATIAL 4C ROBOTIC MECHANISM

A spatial 4C robotic mechanism has four cylindrical joints, each joint permitting relative rotation and translation along a line (see fig. 12). The frame's axes are color coded red, green, and blue to correspond with the local XYZ axes. The link parameters that define the mechanism are listed in table 1 and the joint variables are defined in table 2.

Table 1. Common Normal & Link Parameters of the 4C Mechanism

Link	*Dual Angle*	*Twist*	*Length*
Driving	$\hat{\alpha}$	α	a
Coupler	$\hat{\eta}$	η	h
Driven	$\hat{\beta}$	β	b
Fixed	$\hat{\gamma}$	γ	g

The spatial 4C mechanism may be viewed as a combination of two CC dyads. The driving CC dyad has four independent joint variables, referred to as θ, d_1, ϕ and c_1. The driven dyad also has four independent joint variables, ψ, d_2, δ and c_2. When adjoined by the coupler link, the two dyads form a closed chain spatial 4C mechanism with two degrees of freedom. We chose θ and d_1 to be the independent joint variables. Note that ϕ and c_1 as well as the driven dyad's joint variables are now explicit functions of θ and d_1 and these functions are found below.

4C Robotic Mechanism Analysis

We now present the equations that define the relative movement for the links of a spatial 4C mechanism given its physical dimensions and the input variables, θ and d_1. The closed chain vector loop equations were solved to yield the following equations, [9] and [3].

The coupler angle ϕ is a function of the input angle θ (see eqs. 15).

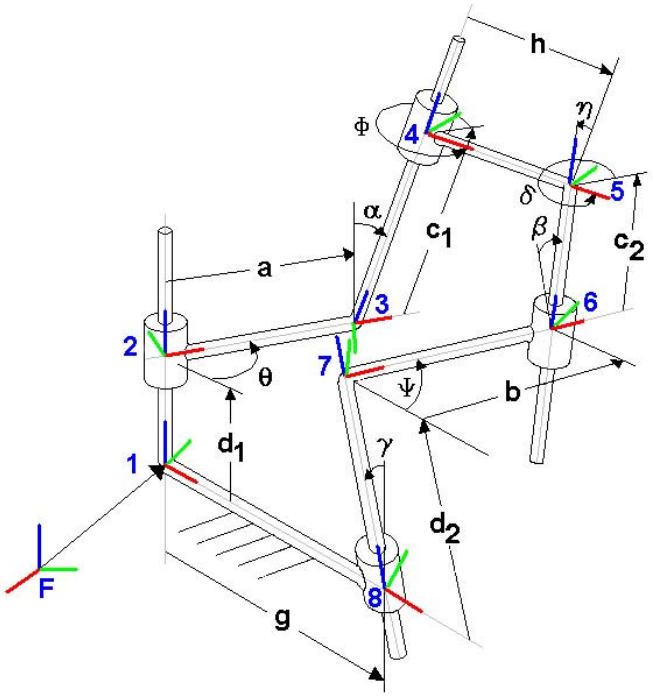

Figure 12. 4C Spatial Mechanism

Table 2. Moving Axes & Joint Variables of the 4C Mechanism

Joint Axis	*Dual Angle*	*Rotation*	*Translation*
Fixed	$\hat{\theta}$	θ	d_1
Driving	$\hat{\phi}$	ϕ	c_1
Coupler	$\hat{\delta}$	δ	c_2
Driven	$\hat{\psi}$	ψ	d_2

$$\phi(\theta) = \arctan\left(\frac{B}{A}\right) \pm \arccos\left(\frac{C}{\sqrt{A^2+B^2}}\right) \tag{15}$$
$$A = \sin(\eta)\sin(\gamma)\cos(\alpha)\cos(\theta) - \sin(\alpha)\sin(\eta)\cos(\gamma)$$
$$B = -\sin(\eta)\sin(\gamma)\sin(\theta)$$
$$C = \cos(\beta) - \cos(\eta)\sin(\alpha)\sin(\gamma)\cos(\theta) - \cos(\alpha)\cos(\eta)\cos(\gamma)$$

Note that ϕ has two solutions corresponding to the two assemblies or circuits of the mechanism.

The output angle ψ is a function of the input angle θ and the coupler angle ϕ (see eqs. 16).

$$\psi(\theta,\phi) = \arctan\left(\frac{B}{A}\right) \tag{16}$$
$$A = \frac{1}{-\sin(\beta)}\Big\{\cos(\eta)(\cos(\alpha)\sin(\gamma) - \cos(\gamma)\cos(\theta)\sin(\alpha)) - \sin(\eta)\cos(\phi)(\cos(\alpha)\cos(\gamma)\cos(\theta) + \sin(\alpha)\sin(\gamma)) + \sin(\eta)\cos(\gamma)\sin(\phi)\sin(\theta)\Big\}$$
$$B = \frac{1}{\sin(\beta)}\Big\{\cos(\eta)\sin(\alpha)\sin(\theta) + \sin(\eta)\cos(\theta)\sin(\phi) + \sin(\eta)\cos(\alpha)\cos(\phi)\sin(\theta)\Big\}$$

The output coupler angle δ, i.e. the angle between the coupler and driven crank is a function of the input angle θ and the output angle ψ (see eqs. 17).

$$\delta(\theta,\psi) = \arctan\left(\frac{B}{A}\right) \tag{17}$$
$$A = \frac{1}{\sin(\eta)}\Big\{\cos(\alpha)\big(\cos(\gamma)\sin(\beta) + \cos(\beta)\sin(\gamma)\cos(\psi)\big) - \sin(\alpha)\cos(\theta)\big(\cos(\beta)\cos(\gamma)\cos(\psi) - \sin(\beta)\sin(\gamma)\big) - \sin(\alpha)\cos(\beta)\sin(\theta)\sin(\psi)\Big\}$$
$$B = \frac{1}{-\sin(\eta)}\Big\{\cos(\alpha)\sin(\gamma)\sin(\psi) + \sin(\alpha)\sin(\theta)\cos(\psi) - \sin(\alpha)\cos(\gamma)\cos(\theta)\sin(\psi)\Big\}$$

The driving coupler translation c_1, the translation along the driving axis, is a function of θ, ψ, δ and the input translation d_1 (see eqs. 18).

$$c_1(\theta,\psi,\delta,d_1) = \frac{A}{B} \tag{18}$$
$$A = d_1\sin(\gamma)\sin(\psi) + a\cos(\theta)\cos(\psi) + a\cos(\gamma)\sin(\theta)\sin(\psi) + h\cos(\delta) - b - g\cos(\psi)$$
$$B = \sin(\eta)\sin(\delta)$$

The driven coupler translation c_2, the translation along the driven axis, is a function of θ, ϕ, ψ and the driving coupler translation c_1 (see eqs. 19).

$$c_2(\theta,\phi,\psi,c_1) = \frac{A}{B} \tag{19}$$
$$A = h\cos(\phi)\cos(\theta) + c_1\sin(\alpha)\sin(\theta) + a\cos(\theta) - h\cos(\alpha)\sin(\phi)\sin(\theta) - g - b\cos(\psi)$$
$$B = \sin(\beta)\sin(\psi)$$

Finally, the translation along the driven axis d_2 is a function of θ, ϕ, ψ, the driving coupler translation c_1 and the driven coupler translation c_2 (see eqs. 20).

$$d_2(\theta,\phi,\psi,c_1,c_2) = \frac{A}{B} \tag{20}$$
$$A = h\cos(\phi)\sin(\theta) - c_1\cos(\theta)\sin(\alpha) + a\sin(\theta) + h\cos(\alpha)\cos(\theta)\sin(\phi) - b\cos(\gamma)\sin(\psi) + c_2\cos(\beta)\sin(\gamma) + c_2\cos(\gamma)\cos(\psi)\sin(\beta)$$
$$B = -\sin(\gamma)$$

MECHANISM COLLISION TESTING

Here we adapt the general methodologies of section 2 to a study of spatial closed chain mechanisms, specifically, the spatial 4C robotic mechanism.

Part 1: Analyzing Mechanism Via Points

Our implementation of the collision detection algorithm presented here requires a set of via points, the constant parameters (α, β, γ, η, a, b, g, h) and the radii of each of the links of a spatial 4C mechanism. The set of via points contains the input values for θ, d_1, number of incremental steps to the next via point, and which solution to use for ϕ. From this given information several tests can be performed to see if the mechanism is unsatisfactory.

Each via point's theta value can be tested to make sure that it is within the allowable motion range of the mechanism. The allowable motion range can be calculated using the link twist angles of the mechanism (see eqs. 21) [14].

$$C_1 = \frac{\cos(\eta-\beta) - \cos(\alpha)\cos(\gamma)}{\sin(\alpha)\sin(\gamma)} \tag{21}$$
$$C_2 = \frac{\cos(\eta+\beta) - \cos(\alpha)\cos(\gamma)}{\sin(\alpha)\sin(\gamma)}$$
$$-1 < C_1, C_2 < 1$$

This gives four possible cases of solutions sets (see fig. 13):

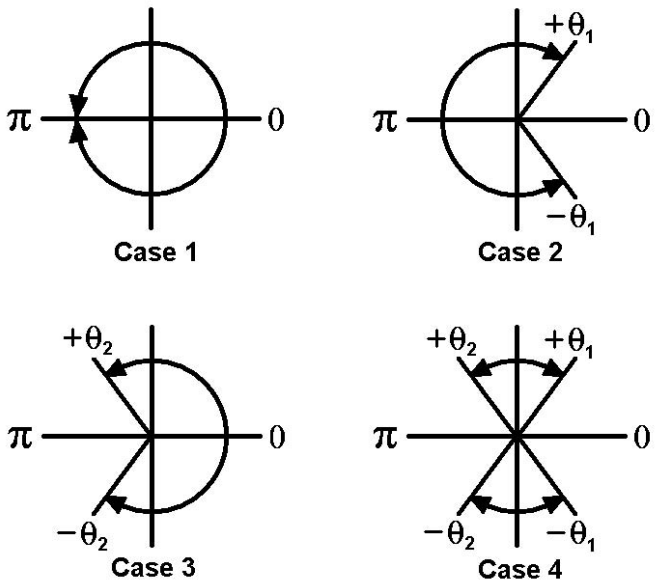

Figure 13. Input Theta Range

Case 1. Neither C_1 nor C_2 are within the allowable range so the input link is capable of full rotation.
Case 2. Only $\theta_1 = \arccos(C_1)$ exists, then the input link rocks across π from $\pm\theta_1$.
Case 3. Only $\theta_2 = \arccos(C_2)$ exists, then the input link rocks across 0 from $\pm\theta_2$.
Case 4. Both $\theta_1 = \arccos(C_1)$ and $\theta_2 = \arccos(C_2)$ are within the allowable range. The input link can rock in two ranges, between θ_1 and θ_2 and $-\theta_1$ and $-\theta_2$ not passing through 0 or π.

Using the above equations the allowable ranges for θ can be determined. If any of the via point's θ values are not in the allowable range, the mechanism is not satisfactory.

In addition, each via point's d_1 can be checked for a sign change or if it approaches zero. Currently, the spatial 4C mechanisms designed by SPADES and VRSpatial use the common normal to connect each link's collar to its axis. If the sign of d_1 changes or if it nears zero a collision will occur between the driving link's collar and the fixed link's common normal.

Similarly, the ϕ solution set can be inspected. The ϕ solution set is passed in as part of the via points. ϕ has two solution sets and the remaining calculations are based on only one of the sets for the mechanism to be acceptable. If the set changes, the mechanism changes circuits and/or moves through a singular configuration.

Part 2: Infinite Line (Cylinder) Generation

The first step in testing the mechanism for a possible collision is generating the Plücker coordinates of its axes at each

Table 3. Mechanism Axes Parameters

Frame	*Axis*	*Rotation*	*Translation*
${}^1_2\mathrm{T}$	z	θ	d_1
${}^2_3\mathrm{T}$	x	α	a
${}^3_4\mathrm{T}$	z	ϕ	c_1
${}^1_8\mathrm{T}$	x	γ	g
${}^8_7\mathrm{T}$	z	ψ	d_2
${}^7_6\mathrm{T}$	x	β	b
${}^6_5\mathrm{T}$	z	δ	c_2

incremental step. We assign right-handed frames to the mechanism that translate and rotate along and about only the local X and Z axes (see fig. 12). The frames are attached at the intersection of the link segments and are aligned such that either its X or Z axis is collinear with the link's direction vector. This kinematic analysis uses standard homogeneous transformations that are translations or rotations with respect to a single local axis (X or Z). Each homogeneous transformation contains the point on the line ($\vec{p}$) and its direction vector (either $\vec{X}$ or $\vec{Z}$).

$$ {}^{i-1}_{i}T = \left[\begin{array}{c|c} \vec{x}\ \vec{y}\ \vec{z} & \vec{p} \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \tag{22} $$

The closed chain mechanism is modelled as two open kinematic chains fixed at frame 1 that are linked by the coupler's common normal segment h. Note that the distance between ${}^1_4\vec{p}$ and ${}^1_5\vec{p}$ is h) (see eqs. 23). The Plücker coordinates of the moving axes are obtained from the matrices ${}^1_4T$ and ${}^1_5T$:

$$ \begin{aligned} {}^1_4T &= {}^1_2T^2_3T^3_4T \\ &= Z(\theta,d_1)X(\alpha,a)Z(\phi,c_1) \\ {}^1_5T &= {}^1_8T^8_7T^7_6T^6_5T \\ &= X(\gamma,g)Z(\psi,d_2)X(\beta,b)Z(\delta,c_2) \end{aligned} \tag{23} $$

where,

$$ [X(\theta,x)] = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ [Z(\theta,z)] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Each frame is located by the homogeneous transformation matrices in space. This permits the use of line geometry and dual number algebra to calculate the distance between the infinite length axes to determine if the mechanism may have experienced a collision. Any axes pair that a collision may have occurred in is written to a file. The file contains the information necessary to test the axes pair that will be modelled and tested as finite length cylinders. The data necessary for identifying the test to be performed is the input values of d_1, θ and the segment numbers that may have collided. Section 5 discusses techniques for minimizing the number of tests required at each step to test completely for self collisions.

Part 3: Determining Maximum Length of each link's moving axis

The links of a spatial 4C robotic mechanism can be modelled by a closed chain of eight line segments defined by twelve points (see fig. 14). Each of the four links are described by three points: one at the center of the link's *collar*, the *elbow* at the intersection of the link's common normal with its axis, and the *end* at the opposite end of its axis. Although each link's collar is co-linear with the previous link's axis, separate points are required for it and for the end of the moving axis.

We assume that each link axis is rigid and that its length will be sized accordingly. Hence, it is necessary to find the maximum length of each link's axis for the desired motion. To do this we use linear interpolation of the θ and d_1 via points to yield a discretized representation of the desired motion. At each discrete point we perform a kinematic analysis of the mechanism via Section 3. Finally, the minimum and maximum values of c_1, d_2, and c_2 are identified. These lengths are used to define the lengths of the finite cylinders that are used to model the link axes for collision detection.

Part 4: Finite Line (Cylinder) Generation

The next step in testing the mechanism is generating the segment endpoints for each of the possible collision segments saved and identified during the infinite length cylinder testing. For each linear interpolation of θ and d_1 of the mechanism, the endpoints of any segment can be generated by using the above kinematic analyses. The end of each link has to be generated differently for each chain (see fig. 15). For the driving chain we substitute d_{1max} for d_1 and c_{1max} for c_1 to obtain the three dimensional coordinates of the end point (see eqs. 24 and 25). However, the driven chain measures the translations in the opposite direction

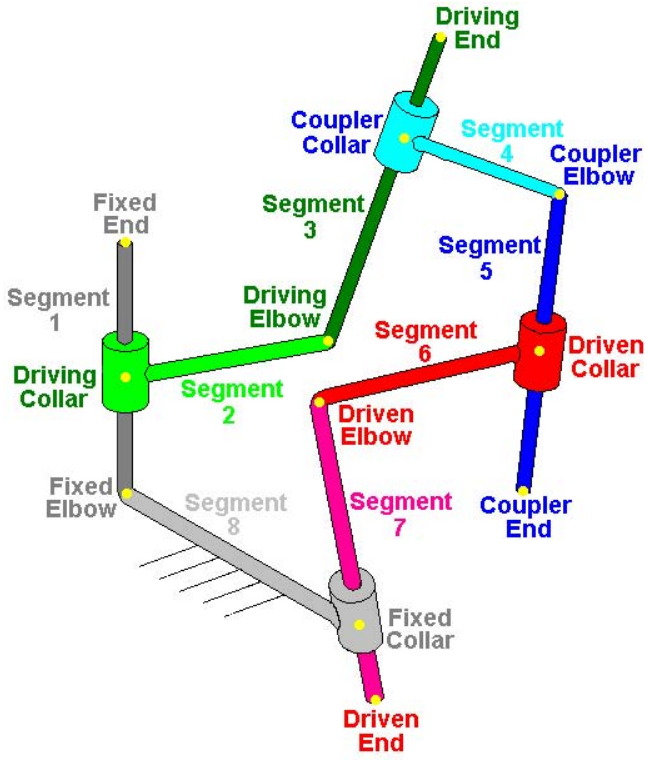

Figure 14. Point Designation

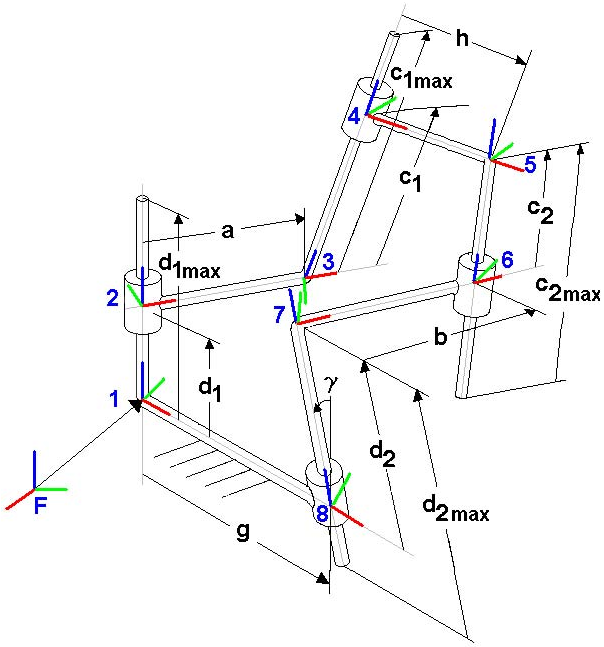

Figure 15. Link Length Definitions

Table 4. Segment Designations of the Spatial 4C Mechanism

Segment No.	*Start Point*	*End Point*
1	Fixed Elbow	Fixed End
2	Driving Collar	Driving Elbow
3	Driving Elbow	Driving End
4	Coupler Collar	Coupler Elbow
5	Coupler Elbow	Coupler End
6	Driven Collar	Driven Elbow
7	Driven Elbow	Driven End
8	Fixed Collar	Fixed Elbow

from collar to elbow instead of from elbow to collar. This results in the end point being in the opposite direction of the elbow relative to the collar (except when the translation is at its maximum/minimum). This makes the translation of the point, for example, c_2 - c_{2max} (see eq. 25).

$$\begin{bmatrix} DrivingEnd \\ 1 \end{bmatrix} = {}^{1}_{4max}T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{24}$$

$$= Z(\theta, d_1)X(\alpha, a)Z(\phi, c_{1max}) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} CouplerEnd \\ 1 \end{bmatrix} = {}^{1}_{5max}T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{25}$$

$$= X(\gamma, g)Z(\psi, d_2)X(\beta, b)Z(\delta, c_2 - c_{2max}) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

For each incremental step the twelve points can be calculated and the eight line segments generated. The segments are numbered starting with the fixed link's axis, proceeding around the closed chain, and ending with the fixed link's common normal (see tbl. 4). This yields the set of line segments for the incremental step that can then be tested to see if a collision has occurred.

Table 5. Case Study - Link Parameters

Dual Angle	*Twist (degrees)*	*Length (unit)*
$\hat{\alpha}$	$\alpha = 65$	$a = 100$
$\hat{\beta}$	$\beta = 35$	$b = 80$
$\hat{\gamma}$	$\gamma = 45$	$g = 70$
$\hat{\eta}$	$\eta = 30$	$h = 90$

Table 6. Case Study - Motion Input

θ *(degrees)*	d_1 *(unit)*	*Increments*	$\pm\phi$
6	100	50	+
-12	80	50	+
-27	100	60	+
-8	110	60	+
3	60	30	+

CYLINDER TESTING LOGIC

It is necessary that for each incremental step every cylinder is tested for possible collisions. Since speed is important for the calculations, we look to reduce the number of tests that must be run for each incremental step. First, cylinders can not collide with themselves since they are rigid links. Testing cylinder 1 for a collision with cylinder 5 is redundant to checking cylinder 5 to cylinder 1. This greatly reduces the number of tests required. The number can be further reduced by observing the design of the mechanism. In a spatial 4C robotic mechanism it is not possible for a cylinder to collide with it's adjacent cylinders. This reduces the number of possible cylinder combinations in a spatial 4C robotic mechanism to twenty.

CASE STUDY

To demonstrate the methodology presented in this paper we used the spatial 4C mechanism described in Table 5. The mechanism's fixed link is grey, driving link green, driven link red and coupler link blue (see fig. 16). Each link's common normal was assigned a radius of five units and each axis a radius of twenty. The axis was modelled as having a larger radius to account for the collar that has to translate and rotate about the link's axis.

A set of via points (see tbl. 6) was then entered for the mechanism and some initial testing was performed. From the link twist values the allowable range of θ was calculated (see eqs. 21) and only C_2 exists (case 3). This made the allowable θ range $78.863 \leftrightarrow -78.863$, rocking across 0 (see eqs. 26).

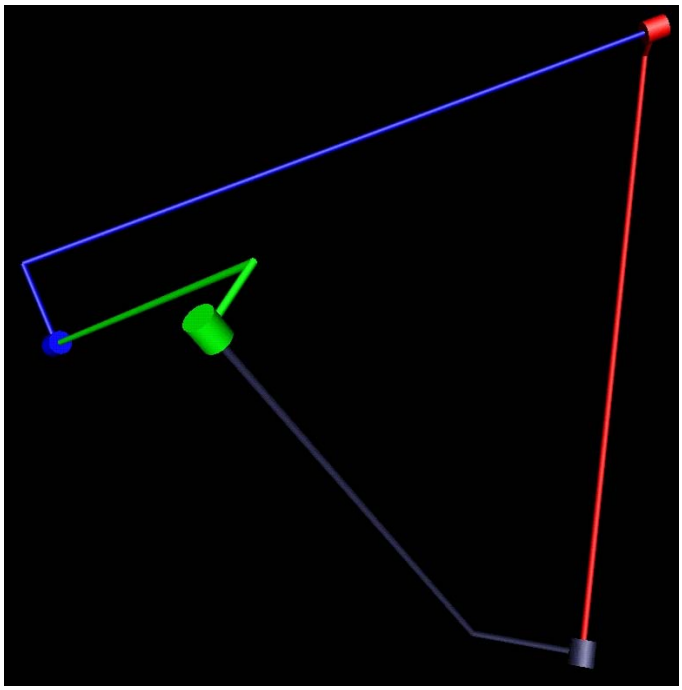

Figure 16. Case Study: 4C Mechanism

$$C_1 = 1.0882 \Longrightarrow \theta_1 \text{ does not exist} \qquad (26)$$
$$C_2 = 0.19315 \Longrightarrow \theta_2 = 78.863$$

After calculating the allowable θ range, each of the via points was tested to make sure that they were within the same allowable range.

The next step in testing the mechanism is to begin incrementally moving the mechanism through it's desired motion. At each step, the first level of collision testing is performed using infinite length cylinders that are calculated and tested for possible collisions. If a possible collision is detected, the data that describes the mechanism's position and which lines may have collided are written to a file. This case resulted in 767 instances when the infinite length cylinders collided. Also during the incremental movement the global translational minimums and maximums of the mechanism are determined (see tbl. 7). Inspection of the mechanism's translations shows that there were no sign changes in the individual translations and that none of them approach zero.

Next, the second level of testing, using finite length cylinders, is performed on each of the possible collisions written to the file. The result is that there was a collision detected. The collision occurred between segments three and six when $\theta = -2.28$ and $d_1 = 90.80$ (see tbl. 8). Figure 17 shows the mechanism when it is in its collision configuration.

Table 7. Case Study - Translation Output

Translation	*Min (unit)*	*Max (unit)*
d_1	60.000	110.000
c_1	122.344	234.576
d_2	16.205	190.978
c_2	34.786	190.747

Table 8. Case Study - Results

Distance (unit)	*Seg. No.*	*Seg. No.*	θ *(degree)*	d_1 *(unit)*
-0.241	3	6	-2.28	90.80

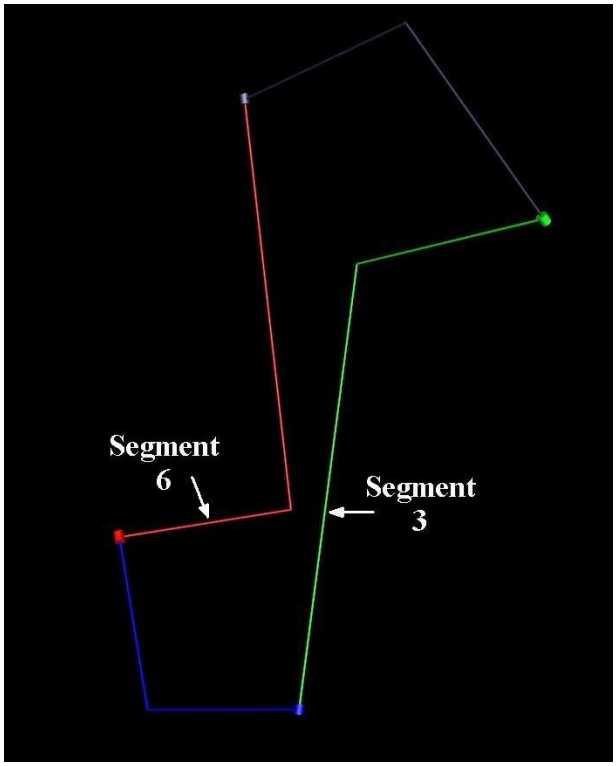

Figure 17. Case Study: Collision Configuration

CONCLUSIONS

In this paper we have presented a novel methodology for detecting collisions of cylindrically shaped rigid bodies moving in three dimensions. This algorithm uses line geometry and dual number algebra to exploit the geometry of cylindrical objects to facilitate the detection of collisions. First, the rigid bodies are modelled with infinite cylinders and an efficient necessary condition for collision is evaluated. If the necessary condition is not satisfied then the two bodies do not collide. If the necessary condition is satisfied then a collision between the bodies may occur and we proceed to the next stage of the algorithm. In the second stage the bodies are modelled with finite cylinders and a definitive necessary and sufficient collision detection algorithm is employed. The result is a straight-forward and efficient means of detecting collisions of cylindrically shaped bodies moving in three dimensions. This methodology has applications in spatial mechanism design, robot motion planning, workspace analysis of parallel kinematic machines such as Stewart-Gough platforms, nuclear physics, medical research, computer graphics and well drilling. A case study examining a spatial 4C robotic mechanism for self collisions was included.

ACKNOWLEDGMENTS

This work was made possible by NSF Grants Nos. #9816611 and #0422705. The collaborations with and contributions of Prof. Vance and her student Denis Dorozhkin of Iowa State University are gratefully acknowledged.

REFERENCES

[1] Belta, Calin and Kumar, Vijay (2000), An efficient, geometric approach to rigid body motion interpolation, *Proceedings of the ASME 2000 Design Engineering Technical Conferences and Computers and Information Conference, DET2000/MECH-14216*.

[2] Dees, S.L. (2001), *Spatial mechanism design using an svd-based distance metric*, Master's Thesis, Florida Institute of Technology.

[3] Duffy, J. (1980), Analysis of mechanisms and robot manipulators, John Wiley & Sons.

[4] Fischer, Ian, (1999), Dual-number methods in kinematics, statics and dynamics, CRC Press LLC.

[5] Fischer, Ian (1994), Novel methods in the displacement analysis of spatial mechanisms, *ASME*.

[6] Fuhrmann, Artur and Schomer, Elmar (2001), A general method for computing the reachable space of mechanisms, *Proceedings of the ASME 2001 Design Engineering Technical Conferences and Computers and Information Conference, DET2001/DAC-21057*.

[7] Kihonge, J., Vance, J. and Larochelle, P. (2002), Spatial mechanism design in Virtual Reality with networking, *ASME Journal of Mechanical Design*, vol. 124, no. 1, pp. 435-440.

[8] Larochelle, P. (2000), Circuit and branch rectification of the spatial 4c mechanism, *Proceedings of the ASME 2000 Design Engineering Technical Conferences and Computers and Information Conference, DET2000/MECH-14053*.

[9] Larochelle, P. (1998), Spades: software for synthesizing spatial 4c mechanisms, *Proceedings of the ASME 1998 Design Engineering Technical Conferences and Computers and Information Conference*.

[10] Larochelle, P. (1994), *Design of cooperating robots and spatial mechanisms*, PhD Dissertation, University of California, Irvine.

[11] Larochelle, P., Vance, J. and Kihonge, J. (2002), Interactive visualization of line congruences for spatial mechanism design, *ASME Journal of Computers and Information Science in Engineering*, vol. 2, no. 3, pp. 208-213.

[12] McCarthy, J.M., (2000), Geometric design of linkages, Springer-Verlag New York.

[13] Mi, Z., Yang, J., Abdel-Malek, K. and Jay, L. (2002), Planning for kinematically smooth manipulator Trajectories, *Proceedings of the ASME 2002 Design Engineering Technical Conferences and Computers and Information Conference, DET2002/MECH-34325*.

[14] Murray, A.P., and Larochelle, P. (1998), A classification scheme for planar 4r, spherical 4r, and spatial rccc linkages to facilitate computer animation, *Proceedings of the ASME 1998 Design Engineering Technical Conferences and Computers and Information Conference, DET1998/MECH-5887*.

[15] Serre, P., Riviere, A., Duong, A. and Ortuzar, A. (2001), Analysis of a geometric specification, *Proceedings of the ASME 2001 Design Engineering Technical Conferences and Computers and Information Conference, DET2001/DAC-21123*.

[16] Stocco, Leo (2001), Path verification for unstructured environments and medical applications, *Proceedings of the ASME 2001 Design Engineering Technical Conferences and Computers and Information Conference, DET2001/DAC-21128*.

[17] Xavier, Patrick (2000), Implict convex-hull distance of finite-screw-swept volumes, *Proceedings of the 2002 IEEE International conference on robotics & automation*.

[18] Zsombor-Murray, P.J. (1992), Spatial visualization and the shortest distance between two lines problem, *Proceedings of the 5th ASEE International Conference ECGDG*. Melbourne, Australia.

[19] Cohen, J., Lin, M., Manocha, D., and Ponamgi, M. (1995), I-collide: An interactive and exact collision detection system for large-scale environments, *Symposium on interactive 3D graphics*, 1995.

[20] Caselli, S., Reggiani, M., and Mazzoli, M., Exploiting advanced collision detection librairies in a probabilistic motion planner.

[21] Hudson, T., Lin, M., Cohen, J., Gottschalk, S., and Manocha, D., (1997), V-collide: Accelerated collision detection for vrml, *Proceedings of the* 2^{nd} *Symposium on VRML*, ACM Press, 1997

[22] T. Johnson and J. Vance (2001), The use of voxmap pointshell method of collision detection in virtual assembly methods planning, *Proceedings of the ASME 2001 Design Engineering Technical Conferences and Computers and Information Conference, DET2001/DAC-21137*.